PATENT ABSTRACTS OF JAPAN

(11)Publication number : 06-274608
(43)Date of publication of application : 30.09.1994

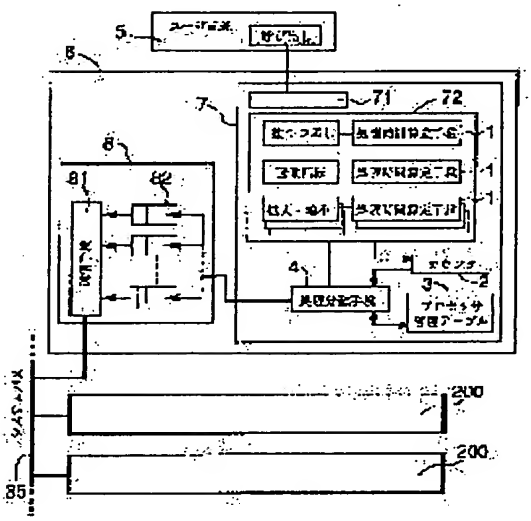
(51)Int.Cl. G06F 15/66
G06F 15/16

(21)Application number : 05-064344
(22)Date of filing : 23.03.1993

(71)Applicant : SEIKO EPSON CORP
(72)Inventor : NAGASAKA FUMIO

(54) MULTIPROCESSOR IMAGE PROCESSOR

(57)Abstract:
PURPOSE: To provide processing utilizing ability at a maximum by optimizing the load of processing partially charged by respective processors at the image processor using plural processors.
CONSTITUTION: An image processing library 72 provided with the procedures of basic operations concerning image processing is equipped with a processing time calculating means 1 for each kind of processing. A virtual mechanical process 6 to perform image processing executes library calling in an estimation mode before real image processing and estimates processing time in advance concerning each processing unit. This value is stored in a counter 2. Real processing distribution to processors is performed by a processing distributing means 4 corresponding to this estimated value of processing time so as to equalize the load of respective processors. The load partially charged by respective processors is recorded in a processor managing table 3 as the cumulative value of this estimated value. The processing distributing means 4 manages load distribution to respective processors and processing result reception and as the entire processor, processing efficiently utilizing the ability of respective processors is provided.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平6－274608

(43)公開日　平成６年(1994)９月30日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　15/66		Ｋ　8420－5Ｌ		
15/16		Ｔ　7429－5Ｌ		

審査請求　未請求　請求項の数１　ＯＬ　（全 20 頁）

(21)出願番号　特願平5－64344

(22)出願日　平成５年(1993)３月23日

(71)出願人　000002369
セイコーエプソン株式会社
東京都新宿区西新宿２丁目４番１号

(72)発明者　長坂　文夫
長野県諏訪市大和３丁目３番５号　セイコーエプソン株式会社内

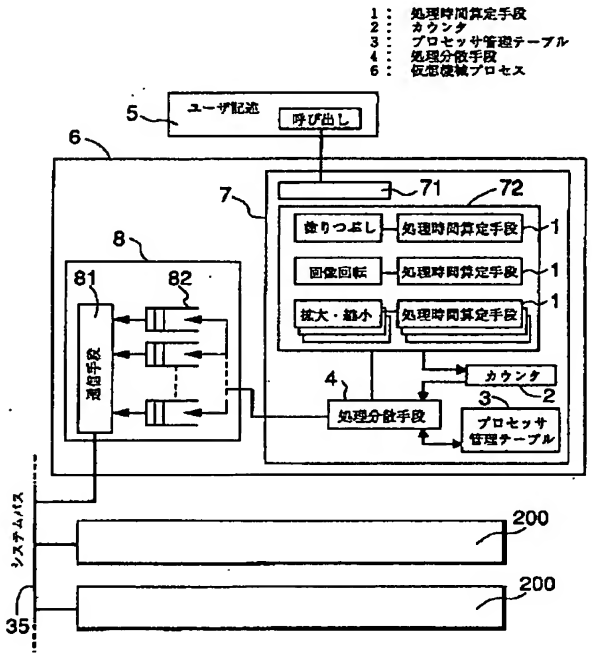
(74)代理人　弁理士　鈴木　喜三郎　（外１名）

(54)【発明の名称】　マルチプロセッサ画像処理装置

(57)【要約】

【目的】　複数プロセッサを用いた画像処理装置において、個々のプロセッサに分担させる処理の負荷の最適化を図り、能力を最大限利用した処理を実現する。

【構成】　画像処理に関する基本操作の処理手順を含む画像処理ライブラリ７２が、個々の処理毎に処理時間算定手段１を持つ。画像処理を行なう仮想機械プロセス６は、実際の画像処理に先だって見積りモードでライブラリ呼び出しを実行し、処理単位について予め処理時間の見積りを行なう。この値はカウンタ２に記録される。実際のプロセッサへの処理分散は処理分散手段４がこの処理時間の見積り値により各プロセッサの負荷が均等になる様に行なう。各プロセッサの負荷分担は、この見積り値の累積値としてプロセッサ管理テーブル３に記録される。処理分散手段４は各プロセッサへの負荷配分、処理結果受信を管理し、処理装置全体としては複数のプロセッサの能力を効率よく利用した処理が実現される。



1

【特許請求の範囲】
【請求項１】　処理対象の画像に関する一連の操作を複数の処理単位に分割して、各処理単位を複数個のプロセッサに分散し、並列に実行することによって処理速度の向上を図る装置であって、
画像処理に関する所定の処理要求を、複数個のプロセッサの中から選択した一つのプロセッサにそれぞれ分散する処理分散手段と、
予め定めた特定の画像操作からなる実行時ライブラリと、
前記実行時ライブラリ中の操作の一つ一つに対応した処理負荷の見積りを行う見積り手段と、によって構成され、目的処理中に前記実行時ライブラリに含まれる画像操作が呼び出される時、操作開始に先立ち、負荷見積り手段が処理負荷を計算し、この値を処理分散手段に通知し、処理分散手段は前記の通知された値から処理単位を分散するプロセッサを決定する事を特徴とするマルチプロセッサ画像処理装置。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】この発明は、複数のプロセッサ間で通信を行なうことによって処理の分散、および並列実行、同期を実現し、所定の処理を行なうマルチプロセッサ装置に関する。特に画像処理技術に応用されるマルチプロセッサ装置の個々のプロセッサへの最適な負荷分散を行なう技術に関連する。
【０００２】
【従来の技術】以下の文中において、「プロセッサの負荷」の意味を明らかにするため、説明を加える。
【０００３】与えられた処理をプロセッサが行なう時、処理開始から完了までに、のべ何マシンサイクルのプロセッササイクルを必要とするかによって、対象処理の処理時間が決められる。通常、稼働中のプロセッサは停止することなく、連続して、１マシンサイクル毎に１個ないし数個の機械語命令を実行し続けるものであるから、機械語実行という視点から見て、プロセッサに（電気的な意味での）負荷の変動が起きる事は無い。
【０００４】しかし、プロセッサを応用した情報処理装置には、複数プログラムの実行や、入出力装置のプログラム間の共用、あるいは周辺装置の資源としての最適利用などを目的とするオペレーティングシステムを持つものが有る。これら特定の情報処理装置（特にコンピュータ装置）において、動作時間はオペレーティングシステムが何らかの処理要求を受け付けて、この目的処理のためのプログラムを実行する時間と、次の処理要求を受け取るまでの待機時間、あるいは並行して実行する他の処理へ切り替えるためのプログラム切り替え処理時間などの総和である。
【０００５】今、待機時間およびオペレーティングシステムの動作時間（プログラム切り替え処理時間など）に

2

比較して、「要求される処理」の処理時間が長大である時、この装置の負荷が大きいと表現する。
【０００６】プロセッサ装置の負荷が大きい場合、新たに別の処理要求を投入しても、これが処理され結果の得られるまでの時間は長くなる不都合が生じる場合が多い。極端にプロセッサ装置の負荷が高まった場合は、入出力装置などの周辺装置を有効に利用するためのオペレーティングシステムの処理にも遅滞を生じ、周辺装置の性能を有効に利用できない不都合が生じる。特にこの傾向はマイクロプロセッサ装置を使用したワークステーション、パーソナルコンピュータ等において顕著である。
【０００７】さらに従来の技術について説明する。
【０００８】画像処理の応用分野に、ワークステーションあるいはパーソナルコンピュータを用いる場合、高密度、多諧調のデータに対しては、現在のマイクロプロセッサ装置の処理速度は充分とは言えない。処理速度不足の問題点を改善する目的で発明された従来技術として、「目的処理を複数の処理部分に分割して複数個のプロセッサを用いて並列処理する事によって処理速度の高速化を図る」技術がある。
【０００９】この技術では各プロセッサにかかる負荷が最適となるように、処理部分が分割されて各プロセッサに配置された時、最も処理速度が向上すると考えられる。今、マルチプロセッサ装置を構成する各プロセッサの処理能力がおよそ同等であるとすれば、各プロセッサに均等な負荷がかかった状態が最適負荷配分であると考える事ができる。
【００１０】もし、処理を複数の部分に分割して個々のプロセッサに配置する以前に、各処理部分がどの程度の負荷をプロセッサに与えるか知ることができれば、負荷がほぼ均等になる様に処理を分散する事は容易である。また、分散する個々の「処理単位」の必要処理時間がそれぞれ小さい場合、負荷配置（スケジューリング）は容易である。この様な場合は「処理単位が細粒度である。」と呼び、この反対の状態を「処理単位が粗粒度である。」と呼ぶ。粗粒度の処理を最適に負荷配置する方法は、あるプロセッサに「必要処理時間の大きい処理」を連続して投入してしまう危険性があり、より難しいと言える。
【００１１】汎用の計算処理応用分野を考えると、一般的には任意の処理について、その計算の負荷を事前に知るという手段は、極めて困難である。そのため従来技術は、各プロセッサ装置の処理の待ち行列の長さを観測する等の方法を用いて、先行して実行中の処理の負荷を、言わば外部から観測することで、負荷判定を行なっていた。この方法について述べた従来発明として、特開昭６３－２１１０６０号公報がある。
【００１２】
【発明が解決しようとする課題】しかし、この従来方法は分散する予定の処理の負荷が事前では未知であるた

め、次の処理段階で負荷の最適分散が行なわれるか否かは不定であるという問題点が有った。特に処理対象のプログラムが処理の並列化を粗粒度で行なう場合、待ち行列にある現在の処理要求の負荷が未知のまま「待ち行列の長さ」だけを判断基準として、次の要求を配置する方法は、危険性が大きい。

【００１３】画像処理に限って説明する。

【００１４】ある画像を複数個のプロセッサで並列に処理しようとした時、処理の並列化の代表的な方法は、次の２通りの方法であると考えられる。この内、本発明は（２）の技術を対象とする。

【００１５】（１）画像が画素データの配列として与えられる形式の場合、この画素データの配列を複数の領域に分割して、個々の領域毎に一つのプロセッサを割り当てる方法。

【００１６】（２）画像が図形形状の記述（曲線、多角形など）の集合で与えられる場合、この記述を複数の操作手順に分割し、個々の操作をそれぞれプロセッサに割り当てる方法。

【００１７】（２）の方法では、連続した操作の中で、並列処理することの出来ない操作が存在する。たとえば、特定の領域を塗りつぶす操作では、「領域を指定する操作」と「塗りつぶす操作」を平行して、独立のメモリを持つ別々のプロセッサに分けて操作出来ないことからも想像できる。このために、（２）の方法では処理単位の粒度を小さくすることの出来ない場合が多い。最も典型的な例は、所定の画像領域を回転、拡大、縮小する場合である。これらの操作は、いずれも必要となる処理時間が長い操作である。

【００１８】図１４は、並列実行時に様々な「粒度」を持つ処理単位をプロセッサに分散する場合の問題点を説明する図である。プロセッサが処理要求を受け取る待ち行列４１に、処理要求４２、４３、４４らがエンキュウされている状態を示している。ここで説明のために、各処理単位の必要処理時間を横方向の線分の長さによって示した。また、処理要求とは”複数の操作からなるプログラムの一部分であって、並列処理のためにプロセッサに割り当てられた単位”に対する実行の要求である。図中、第１の待ち行列４１の長さは３であるのに対し、第２の待ち行列４１の長さは５である。図から明らかなように、粒度に大きなばらつきのある場合、待ち行列の長さからプロセッサの負荷の推移を評価することは出来ない。

【００１９】すなわち、従来発明である”各プロセッサの要求受け付け待ち行列の長さから、負荷判断を行ない、処理の分散を行なう方法”は、複数の負荷の大きな処理を特定のプロセッサに集中して配置してしまうことを予防できないという問題点を持つ。従来発明を用いると、あるプロセッサで実際に処理遅れが発生した後に、過負荷であったことが検出されることになる。

【００２０】本発明は、上記の従来発明の問題点を鑑みてなされたものであり、マルチプロセッサによる並列処理を画像処理分野に用いた場合に、各プロセッサへの負荷の最適な分散を実現し、画像処理装置の処理能力を最大限に利用する事を目的としている。

【００２１】

【課題を解決するための手段】このような問題を解決するために本発明のマルチプロセッサ画像処理装置は、処理対象の画像に関する一連の操作を複数の処理単位に分割して、各処理単位を複数個のプロセッサに分散し、並列に実行することによって処理速度の向上を図る装置において、画像処理に関する所定の処理要求を、複数個のプロセッサの中から選択した一つのプロセッサにそれぞれ分散する処理分散手段と、予め定めた特定の画像操作からなる実行時ライブラリと、この実行時ライブラリ中の操作の一つ一つに対応した処理負荷の見積りを行う見積り手段と、によって構成され、目的処理中に前記実行時ライブラリに含まれる画像操作が呼び出される時、操作開始に先立ち、負荷見積り手段が処理負荷を計算し、この値を処理分散手段に通知し、また処理分散手段はここで通知された値から処理単位を分散するプロセッサを決定する事を特徴とする。

【００２２】

【作用】並列処理の応用分野の内、画像処理に限って考えた場合、直線発生、塗りつぶし等の典型的な幾つかの処理は、処理対象領域等の情報を用いて負荷を算定する事が可能である。そこで本発明では、マルチプロセッサによる並列画像処理システムの各画像処理実行プログラムライブラリ毎に処理時間算定手段を用意した。これによって、実際の処理以前に負荷を求める事ができる。

【００２３】処理時間算定の計算は、画素発生の実際処理よりは遥かに軽微であるため、全体の処理を妨げる事は無い。また、事前に処理の負荷を知って処理分散を行なう事で各プロセッサの現在の負荷、処理能力に最適な方法で処理の配置を行なう事が可能となる。

【００２４】

【実施例】本発明に好適な実施例について、以下に図を用いて説明を加える。説明は以下の各節に従って行う。

【００２５】１．本実施例の特徴（従来発明との違い）

１．１　本実施例の構成

１．１．１　装置全体の構成（図４、図３）

１．１．２　オブジェクトコード実行段階の構成（図２）

１．１．３　仮想機械プロセスの構成（図２）

１．１．４　並列実行開始時の仮想機械プロセスの動作概要（図２）

１．２　本実施例の特徴（図１、図５）

２．負荷検出動作

２．１　本実施例の負荷計算手段（図９、図１０）

３．マルチプロセッサ画像処理装置の構成と処理分散

３．１　並列処理の記述方法
３．１．１　並列処理の記述の文法（図１１、図１２、図１３）
３．１．２　組み込み手続きの説明（図１２）
３．２　並列記述の翻訳結果（図１２、図１３、図６、図８）
３．３　処理負荷の見積り動作（図７）
４．他の実施例（図１５）
１．本実施例の特徴（従来発明との違い）
本実施例の構成を明らかにするため、図４、図３、図２、図１の順に説明を行う。処理手順の説明には図５を用いる。本実施例の特徴となる部分は１．２節で述べる。
【００２６】１．１　本実施例の構成
１．１．１　装置全体の構成（図４、図３）
図４は、一つの実施例として挙げるマルチプロセッサ画像処理装置の処理手順の概要を説明した図である。使用者は入力装置３７を用いてエディタ４０１を操作し、プログラム言語によって画像処理内容を記述したソースファイル４０４を作成する。次に使用者は、コンパイラ４０２を実行する。コンパイラ４０２は、ソースファイル４０４を読み込み、プログラム言語による記述内容を翻訳してオブジェクトファイル４０５を出力する。オブジェクトファイル４０５は、所定の書式で書かれた一連の機械語命令によって構成される。本実施例では、特定のプロセッサの機械語を直接実行する方式に代えて、仮想機械方式を用いた。オブジェクトファイル４０５は、この仮想機械の機械語で構成される。仮想機械プロセス４０３は、オブジェクトファイル４０５を読み取り、その機械語命令を実行する。仮想機械プロセス４０３は、命令語として表示処理命令があれば、表示装置３８に画像を出力する。本実施例の仮想機械は、複数のプロセッサを用いて実現される。このための構成を以下に説明する。
【００２７】図３は、一つの実施例として挙げるマルチプロセッサ画像処理装置の全体の構成を示す図である。マイクロプロセッサユニット（以下ＭＰＵ）３１、ＲＡＭ３２、ＲＯＭ３３を実装したプロセッサ基板３４が、システムバス３５によって複数個接続される。入出力周辺制御装置（以下Ｉ／Ｏ装置）３６もシステムバス３５を介して接続される。
【００２８】個々のプロセッサ基板３４には、それぞれオペレーティングシステムが実装される。オペレーティングシステムは、ＲＯＭ３２に格納され、電源立ち上げ後、自動的に実行状態に入る。以後、一つのプロセッサ基板３４は、オペレーティングシステムに管理される独立した処理装置とみなして、プロセッサエレメント２００（ＰＥ２００）と呼ぶ。
【００２９】仮想機械プロセス４０３は、オペレーティングシステムのファイル管理機能を用いて、オブジェクトファイル４０５をＲＡＭ３２に読み込んだ後、個々の機械語命令を実行する。仮想機械プロセス４０３、エディタ４０１等のプログラム、またオペレーティングシステムも作業領域としてメモリ領域を消費する。この領域はいずれもＲＡＭ３２に配置される。個々のメモリの割り当てはオペレーティングシステムが管理する。
【００３０】オブジェクトファイル４０５の内容について説明を補足する。
【００３１】Fortran、Pascal等に代表される一般的なプログラム言語では、文字の印字、データの入力など、ユーザプログラムで一般的に用いられる基本操作を、使用者がその都度書き下す不便を省くことを目的としてこれら基本操作をまとめたライブラリを用意する。ライブラリは、所定の高級言語の文法で使用者が呼び出すことの出来る複数の基本的な操作（サブルーチン、プロシージャ）の集合である。
【００３２】本実施例の記述言語については後述するが、上記に類似したライブラリを用意した。本実施例の記述言語は、応用分野が画像の記述であることから、直線描画、曲線描画、塗りつぶし、画像回転等の画像処理の基本操作をライブラリ化した。使用者がソースファイル４０４を記述した時、必要に応じてライブラリに含まれる関数、プロシージャの呼び出しを記述できる。この結果、オブジェクトファイル４０５には、ライブラリ機能の呼び出しを行なう機械語命令が含まれる。
【００３３】１．１．２　オブジェクトコード実行段階の構成（図２）
図２は本実施例のマルチプロセッサ画像処理装置においてプログラムの実行時の状態を説明した図である。
【００３４】一つのプロセッサエレメント２００の中では、オペレーティングシステム２０４（以下ＯＳ２０４）がプログラムの実行を管理する。ＯＳ２０４が管理するプログラムの実行単位を、これ以後「プロセス」と呼ぶ。「プロセス」は、オペレーティングシステムにおける、プロセッサ資源、メモリ資源割り当ての実行時の単位である。すなわちプロセスは、プロセスの識別子及び実行管理、メモリ管理のための情報を含むプロセスヘッダと、中断の際に現在のプロセッサのレジスタの状態を保存するための領域と、オブジェクトコード領域、スタック領域からなる管理単位である。
【００３５】ＰＥ２００に実装された個々のＭＰＵ３１は、ＯＳ２０４を実行する。ＯＳ２０４は、ＯＳ２０４に組み込まれたスケジューラ２０５の定める手順で、複数の応用プログラムを実行する。すなわち、スケジューラ２０５は、Ｉ／Ｏ装置３６への処理で待機状態に入ったプロセス、または割り当ての微小時間（ここでは５msec）を経過したプロセスを一時的に停止し、スケジューラ待ち行列２０６の末尾に追加する。次にスケジューラ２０５は、スケジューラ待ち行列２０６の先頭にあるプロセスを取り出し、停止状態から実行状態に切り替え

る。プロセスの切り替えは、ＯＳ２０４に組み込まれたサービスプログラム２０７が処理する。この方式によってＯＳ２０４は、同時に複数のプログラムを実行する。

【００３６】使用者は、図４に示したエディタ４０１、コンパイラ４０２、仮想機械プロセス４０３らを、同時にＰＥ２００で実行して良い。図２は、仮想機械プロセス４０３の一つが実行中の状態を図示した。実行中の仮想機械プロセス４０３を、仮想機械プロセス６として図示する。また、他の実行中のプロセスを２０１で示す。

【００３７】

１．１．３　仮想機械プロセスの構成（図２）

仮想機械プロセス６は、処理対象のオブジェクトファイル４０５をメモリ上に読み込み実行する。メモリ上に展開された一つのオブジェクトファイル４０５のデータをユーザ記述５として図示する。また、メモリ上に読み込まれた他のオブジェクトファイル４０５のデータを、オブジェクトコード２０２として図示する。仮想機械プロセス６の構成要素を２つに分けて説明する。一つは実際の仮想機械語を読みとって実行する仮想機械のインタープリタ７である。他の一つは実行時に仮想機械プロセス６の中に確保されるシステムの資源８である。

【００３８】インタープリタ７は命令語の内容を意味理解し、必要な処理への分岐を制御するプログラムインターフェース７１を持つ。また、インタープリタ７は画像処理の個々の操作の処理プログラムからなるライブラリ７２と、処理分散手段４を持つ。プログラムインターフェース７１は、入力バッファからの待ち行列７１１と分岐テーブル７１２を持つ。ユーザ記述５の中にソフトウェア割り込みを引き起こす命令語があると、ＭＰＵ３１の処理番地はプログラムインターフェース７１の所定の番地へジャンプする。インタープリタ７は、ソフトウェア割り込みの種類と引数を検査し、この引数の指定に応じた関数、プロシージャの処理番地を分岐テーブル７１２から決定する。次にインタープリタ７は、この番地へジャンプする。この手順によって、ＭＰＵ３１の処理番地はライブラリ内の関数、プロシージャに移動する。この動作については、図６を用いて後述する。

【００３９】システムの資源８は、プロセッサ間の通信を制御する通信手段８１、これに対する待ち行列８２を持つ。仮想機械プロセス６は、並列実行を行なう各プロセッサ毎に待ち行列８２を作成する。この他システムの資源８は、仮想機械プロセス６の作業メモリ、ファイルからの入力バッファ等、ＯＳ２０４のサービスにより確保した資源を含む。

【００４０】１．１．４　並列実行開始時の仮想機械プロセスの動作概要（図２）

ユーザ記述５の中に、並列実行を引き起こす命令語が有った場合の仮想機械プロセス６動作について説明する。

【００４１】ユーザ記述５に処理の並列化を行なう記述が有った場合、インタープリタ７はこれを解釈し、ＭＰＵ３１の処理番地を処理分散手段４内の処理番地に書き換える（すなわちジャンプする）。ＭＰＵ３１は処理分散手段４を実行し、並列化する処理単位を受け取るＰＥ２００の番号を決定する。続いて、ＭＰＵ３１は処理分散手段４を実行し、並列化する処理単位をＰＥ２００に対して割り当てた待ち行列８２にエンキューする。この後、ＭＰＵ３１は、通信手段８１を実行する。通信手段８１は、システムバス３５の空き時間を調停する装置２０３を制御し、システムバス３５を用いて処理単位を指定したＰＥ２００に転送する。この時、システムバス３５の調停装置２０３の処理には、非同期的な時間の遅延が伴う。そこで、ＭＰＵ３１は通信手段８１の処理に入った後、処理結果を待たず、再びＯＳ２０４の処理番地にジャンプする。ここで、ＭＰＵ３１はスケジューラ２０５の処理を実行し、この仮想機械プロセス６を休眠させ、スケジューラ待ち行列２０６から取り出した次のプロセスの処理番地にジャンプする（これによって次のプロセスが＜再＞走行する）。

【００４２】この結果起動された他のプロセスが、やがて単位時間を経過するか、非同期の時間待ちで休眠するか、いずれかの場合、更に他のプロセスが実行される。この繰り返しの後、再びこの仮想機械プロセス６が実行される。この時、ＯＳ２０４はＭＰＵ３１の処理番地を仮想機械プロセス６内部の前回休眠したプログラム番地に書き換える。これによって先に中断した通信手段８１が再開される。通信手段８１を呼び出したインタープリタ７は、並列実行の処理単位が他のＰＥ２００に転送済みであれば、次の命令語を読み取り、処理を続ける。

【００４３】１．２　本実施例の特徴（図１、図５）

本実施例の特徴を図１によって説明する。図１は、図２の構成の中から、更に本実施例の特徴となる部分を取り出し示した図である。

【００４４】ＭＰＵ３１は、仮想機械プロセス６を実行している時、ユーザ記述５の中に画像処理ライブラリ７２の呼び出しの命令語があると、画像処理ライブラリ７２の中の個々の処理にジャンプする。この結果、ＭＰＵ３１は例えば｛塗りつぶし、画像回転、拡大・縮小｝等の操作を実行する。本実施例では、これら個々の操作は２通りの処理モードを持っている。すなわち、実際に画像処理の操作を行なうモード（実行モード）と、画像処理は行なわず、処理に要する時間を見積もる計算だけを行ない、この結果を内部のカウンタ２に記録するモード（見積りモード）である。この目的から、画像処理ライブラリ７２は、個々の操作毎に処理時間算定手段１を持つ。

【００４５】並列して実行される一つの処理単位は、複数の仮想機械語命令からなるプログラムの記述である。この中に、何回かの画像処理実行時ライブラリ７２の呼び出しが含まれているとき、あらかじめこの処理単位を見積りモードで実行しておけば、この処理結果として、

処理時間の見積りの値がカウンタ２に積算される。処理分散手段４は、このカウンタ２の値を処理分散先のプロセッサの決定に用いる。この時、処理分散手段４はどのＰＥ２００にどの位の規模の処理を配置したか、その累積を記録する目的でプロセッサ管理テーブル３を使用する。

【００４６】処理分散手段４の処理の流れを図５を用いて説明する。

【００４７】処理分散手段４の実行に先だってユーザ記述５の記述内容に従い、仮想機械プロセス６が、プロセス複製Ｓ５０１と、見積りモードでの実行Ｓ５０２を処理する（この処理手順は３節に述べる）。この結果、カウンタ２には、処理時間の見積り値が記録されている。処理分散手段４は、プロセッサ番号を初期設定し（Ｓ５０３）、プロセッサ管理テーブル３を参照して、負荷最小のプロセッサの番号を記録する（Ｓ５０４）。この処理は、プロセッサ番号を更新しつつ（Ｓ５０５）、全てのプロセッサに対して行なわれる。これによって、負荷最小であるプロセッサ番号が決定できるので、このプロセッサ番号を持つＰＥ２００に処理を分散する。すなわちプロセッサ管理テーブル３のこのプロセッサ番号に該当するフィールドに新たな負荷の値を加算し（Ｓ５０６）、処理単位をＰＥ２００に転送する（Ｓ５０７）。

【００４８】以上の構成で、本実施例と従来発明の差であり、本実施例を特徴付けているのは、処理時間算定手段１である。この動作を次に「２．１　本実施例の負荷計算手段」で説明する。

【００４９】２．負荷検出動作

従来技術が待ち行列の長さを負荷判定の指標としていたのに対し、本実施例は、負荷を予め求めた後に、分散先を決定する方法を採る。この点が従来技術と、本実施例の差である。

【００５０】

２．１　本実施例の負荷計算手段（図９、図１０）

処理時間算定手段１の動作について説明する。

【００５１】処理時間算定手段１は、画像処理実行時ライブラリ７２に含まれる処理（プロシージャ）一つにつき、一つずつ対応して組み込まれる。処理時間算定手段１は、対応するプロシージャが呼び出された時、処理開始から完了までにどの程度の処理時間を必要とするか求める手段である。この時求める処理時間は、正確な値である必要はない。また、処理時間算定手段１の目的は、複数個のプロセッサに配置される処理の、それぞれの負荷を知ることであるから、値は相対的な値で良い。更に言えば、処理の規模、複雑さの程度に応じて、比例関係を維持して数値を出力できれば目的を果たすことができる。

【００５２】直線発生処理の場合の負荷の見積りを図９の流れ図を用いて説明する。直線上の画素発生はBresenhamのアルゴリズム（J.E.Bresenham, "Algorithm for Computer Control of a Digital Plotter, "IBM SystemJournal, Vol.4, No.1, 1965）を用いる。この時、画素発生の処理時間は、ほぼ線分の長さに比例する。しかし、線分が画像メモリの配置に対し、水平または垂直である場合、画素発生手順は大幅に簡略化できる。そこで、直線発生の負荷見積りは、線分が水平の場合、線分の長さ×重みＭ１とし（Ｓ９０１）、斜線の場合は線分の長さ×重みＭ２とし（Ｓ９０２）、垂直の場合は線分の長さ×重みＭ３と（Ｓ９０２）した。ここで｛Ｍ１＝１．０、Ｍ２＝９．０、Ｍ３＝１．３｝である。

【００５３】次に画像の回転の場合を説明する。例えば、図１０（ａ）で示した図形１００が、座標軸１０１に対し図示の位置関係に記述されているとする。これを図１０（ｂ）に示した様に、座標軸１０１に対して角度φをなす別の座標軸１０２の系に写像する操作を行なう。本実施例では、「１．１．１　装置全体の構成」に述べた通り、図形を｛直線、曲線｝と言った形状を表す操作とそのパラメータを記述したソースファイル４０４として与える。この方式で回転操作は、「アフィン変換の操作の命令語」とこの引数として与える「行列の各要素」で表される。アフィン変換の操作の対象は、曲線、直線を与える記述（図形を発生する処理関数とその引数）である。すなわち、実質的には、アフィン変換を経て、曲線・直線発生の操作に対して与えるパラメータが変更されるだけである。この結果、本実施例で実際に仮想機械プロセス４０３を実行するプロセッサの負荷は、アフィン変換後に変わった線分の長さに比例する。周知の様にベジエ曲線、円弧、放物線はいずれも最終的に折れ線近似として画素発生を行なう。このため、プロセッサ負荷は、実際に発生すべき直線（折れ線）の長さを求めて、図９と同じ手順で決定できる。この値によるプロセッサ負荷は回転後の図形１００を囲む矩形１０３の面積に比例しない。

【００５４】図形の拡大、縮小もアフィン変換によって行なわれるので、負荷見積り方法は上記と同一である。これに対し、所定領域の塗りつぶし操作は、単純に対象領域の面積に比例する。従って、本実施例では面積を求めることによって負荷を見積る。

【００５５】３．マルチプロセッサ画像処理装置の構成と処理分散

３．１　並列処理の記述方法

本実施例においては、図形発生を発生手順を示す一連の「操作」と、この操作に対する「パラメータ」を与えることによって記述する。ここでプログラム言語の記述に添って言えば、個々の「操作」は”関数”あるいは”プロシージャ”と呼ばれる機構に相当する。また「操作に対するパラメータ」は、関数、プロシージャへの”引数”と見做すことができる。この様な、”図形発生手順のプログラム記述への置き換え”は周知の技術である。この種類の記述言語の典型的なものはページ記述言語と

呼ばれ、ページプリンタの制御に使用される。

【００５６】そこで本実施例はページ記述言語の方法にならい、プログラム言語によって図形発生手順を記述することにした。説明を容易にするため、言語の文法はおよそプログラム言語Pascalの文法に準じている。これに、並列処理を記述する構文、及び複数個の図形記述のための標準手続きを組み込んだ。

【００５７】３．１．１　並列処理の記述の文法（図１１、図１２、図１３）

並列処理部分を記述する文法について説明する。周知のように、プログラム言語のコンパイラ・プログラムを作成する方法は、様々な手段が開発され公知となっている。ここでは文法の説明のため本実施例で用いる図形記述言語の並列記述の構文をバッカス記法に類似の記法で図１１に示した。バッカス記法は言語の構文を”生成規則”と呼ぶ形の文法で記述するものである。図１１は言語全体の構文規則を示したものではなく、並列記述文（cobegin – coend文）およびその周辺の構文規則を示したものである。

【００５８】図１１の構文定義に従って、図１２の様なプログラム例を記述できる。ここでcobegin文１２１からcoend文１２２までが並列実行の対象となる。並列処理の単位は、//記号１２３から次の//記号１２３まで、または//記号１２３から、otherwaize文１２４までである。本実施例のコンパイラは図６のプログラム部分に対し、図１３に示す機械語プログラムを生成する。但し図１３は、実際の機械語のオブジェクトコードを、ディスアセンブルした形式によって記述している。[]内の１６進数数字は、仮想機械の命令語の値である。

【００５９】図１３のプログラムリストの中で、命令語TRAP（図示１３１）について説明する。本実施例のコンパイラの生成するコードでは、ライブラリに含まれる関数及びプロシージャの呼び出しは一種のソフトウェア割り込みによって行う。命令語TRAPは、このソフトウェア割り込みを引き起こす命令語である。仮想機械はこの命令語を実行すると、予め定められた特定のアドレスの値にプログラムカウンタを書き換える。この番地に書かれたプログラムは、TRAP命令の引数を検査し、引数の値に基づきライブラリ中の指定された操作を呼び出す内容である。仮想機械はこのプログラムによって、ライブラリ内の各種関数、プロシージャを呼び出し実行する。

【００６０】１３２、１３３、１３４はライブラリ呼び出しの実装の例である。ここでは、１３２が手続き（またはプロシージャ）line()の呼び出しに対応し、１３３が手続きstroke()に、また１３４が手続きfill()に対応する。図１２のソースコード中に現れるこれら手続きの呼び出し文は、仮想機械語命令のBSR[0d]（サブルーチン呼び出し命令）に翻訳される。引数は、この時の実行番地から、手続きの先頭番地への相対値として与えられる。

【００６１】

３．１．２　組み込み手続きの説明（図１２）

ライブラリの中にどの様な機能を組み込むべきかは、処理系開発の目的と、記述言語の仕様によって決まる事項である。従って、本実施例が挙げる操作は、画像処理の操作の一例に過ぎない。図１２の記述ではライブラリに含まれる操作の中から、｛line()、stroke()、fill()｝を例として用いている。以下これらプロシージャについて説明する。

【００６２】○手続きｌｉｎｅ（ｘ１，ｙ１，ｘ２，ｙ２）

このプロシージャは、４つの整数を引数として呼び出される。上記ｘ１、y1、x2、y2には任意の整数を代入する事ができる。このプロシージャは、現在処理中の線幅を用いて、座標(x1,y1)から座標(x2,y2)まで直線上に画素発生を行なう（この線分は、数学的な線分と異なり、画素密度に応じた線幅を持つ）。

【００６３】

○手続きstroke(N,x1,y1,x2,y2, ... ,xN,yN)

このプロシージャは、可変個数の整数を引数に持つ。但し、引数の整数は２個で一対の座標値として評価され（リスト中の(xi,yi)）、全体の座標の個数は第１の引数（N）として与える。故に、このプロシージャは、全体で2N+1個の整数引数を持つ。このプロシージャは、N個の座標からなる閉じた多角形領域を登録する（この時、折れ線は線幅を持たないとして評価する）。

【００６４】○手続きfill()

このプロシージャは引数を持たず、直前に登録された領域があれば、この領域内部を塗りつぶす。塗りつぶしのパターンおよび、領域の線幅には、この時点でのパターン、線幅を用いる。

【００６５】３．２　並列記述の翻訳結果（図１２、図１３、図６、図８）

図１２のソースコードの中で、特に１２１から１２２に示される並列記述部分がどの様に機械語に翻訳され、実行されるかを図１３と図６を用いて説明する。一つの並列記述文（//文１２３に続く文）は、機械語FORK[2c]に始まりEXIT[2d]に終わる一連の記述に翻訳される。１３５は、文１２３で始まる一行の内容を翻訳した結果の機械語である。すでに述べた通り、この機械語記述は仮想機械プロセス４０３によって実行される。仮想機械プロセス４０３の処理は図６で示す永久ループである。

【００６６】以下仮想機械プロセス４０３の処理を順次説明する。

【００６７】まず命令語がフェッチされる（Ｓ６０１）。この命令語がFORK[2c]であれば、プロセスの複製が行なわれる（Ｓ５０１）。プロセスの複製は、現処理中のユーザ記述５と全く同一のオブジェクトコードをメモリ上に複製し、このオブジェクトコードに関するプログラムカウンタの記録を、現在処理番地に設定する処理

である。この時、FORK[2c]を実行したユーザ記述５の側では、内部フラグを０とする。他方、複製された側は内部フラグを１とする。

【００６８】複製された側のユーザ記述は、転送バッファに置かれ、配置先のＰＥ２００が決定した時点で、処理分散手段４によって転送される。転送先のＰＥ２００では、仮想機械プロセス４０３がこのユーザ記述を受けとって処理を継続する。処理分散手段４は、命令語EVAL[2f]の処理で実行される。これは「３．３　処理負荷の見積り動作」で述べる。

【００６９】本実施例のコンパイラ４０２は、通常FORK[2c]に続いてEVAL[2f]命令を配置する。これを読み取った場合、仮想機械プロセス４０３は、次のプログラム番地をスタックにプッシュし、見積りモード実行のための内部フラグを「真」にする（Ｓ６０２）。見積りモードについては「１．２　本実施例の特徴」でやや述べた。これに続いて実際の見積りモードの実行が行なわれる（Ｓ６０３）。見積りモードの実行が完了すると、見積りモード実行のための内部フラグを「偽」とし、プログラムカウンタの値をスタックから取り出す（Ｓ６０４）。

【００７０】これによって通常の実行状態に戻り、処理が続く。次に実行される命令はBEQ[0c]である。この命令は内部フラグが０のとき分岐し、非０の時、次の番地以降の処理を指定する命令である。前述の様に、処理をFORK[2c]命令で並列化した時、複製元となったユーザ記述では内部フラグ＝０である。従ってFORK[2c]を実行したユーザ記述の側は、分岐して次の実行に入る。すなわち、図１２のソースコードで言えば、//文１２３以後の文はスキップして、次の//文１２３またはotherwise文１２４を処理する。

【００７１】次に、処理分岐の発生元となったＰＥ２００の側での仮想機械プロセス４０３の動作を更に説明する。FORK[2c]とBEQ[0c]命令の実行によって、処理分岐の発生元となったユーザ記述５は、結局Otherwise文１２４からcoend文１２２までの範囲だけを実行する。本実施例のコンパイラ４０２はcoend文１２２に対してWAIT[2e]命令を発生する。仮想機械プロセス４０３は、WAIT[2e]命令が有ると、WAIT処理を行なう（Ｓ６０５）。これは、他の各ＰＥ２００での処理結果が受信バッファの待ち行列に有るか検査し、全てのＰＥ２００から処理結果が戻るまで待機する処理である。

【００７２】処理の流れを図８に示した。仮想機械プロセス４０３は、受信バッファを検査し、処理結果の受信有無を検査する（Ｓ８０３）。受信が無ければ仮想機械プロセス４０３は休眠し（Ｓ８０４）、受信シグナルを受けたＯＳ２０４が、このプロセスを再起動（Ｓ８０５）するまで、他のプロセスが実行される。再起動後、仮想機械プロセス４０３は、処理結果を取得し、処理終了となったプロセッサ番号を検出する（Ｓ８０６）。また処理単位に一意に与えられる処理番号を取り出し、これからプロセッサ管理テーブルの該当するプロセッサの項目にアクセスし、処理番号から処理負荷の値を検索し、この値をプロセッサ負荷から減算しておく（Ｓ８０７）。仮想機械プロセス４０３が、分散した全ての処理単位に対し、この処理を行なった場合（Ｓ８０８）、WAIT処理は完了する。

【００７３】他方、FORK[2c]命令によって、処理分岐された側の動作を説明する。処理分岐の結果、処理分散手段４が、ユーザ記述（複製）を分散先のＰＥ２００に送る。これを受け取ったＰＥ２００上の仮想機械プロセス４０３は、図６の流れ図に添った処理を行なう。但し、内部フラグは'＝１'であり、またプログラムカウンタの値は、FORK[2c]命令の直後の位置にある。内部フラグ＝１のとき、EVAL[2f]命令は何も行なわない。しかし仮想機械プロセス４０３は、他の命令を通常通り処理する。これにより分散先のＰＥ２００では、記述１３５を例にとれば、EXIT[2d]命令までが通常通り処理される。

【００７４】EXIT[2d]命令の処理の流れを図８に示した。仮想機械プロセス４０３は、EXIT[2d]命令を検出した後、処理結果と、処理単位毎に一意に決まる処理番号を送信バッファに置く（Ｓ８０１）。次に仮想機械プロセス４０３は、これを、バス調停装置２０３を用いて、システムバス３５を介して、処理要求元のＰＥ２００に送信する（Ｓ８０２）。

【００７５】図６に戻って説明を続ける。

【００７６】ユーザ記述の終了は、機械語命令HALT[10]に翻訳される。これを検出した仮想機械プロセス４０３は、一つのユーザ記述５について処理を終える。しかし更に仮想機械インタープリタ７の待ち行列７１１に、別のユーザ記述が要求としてエンキュウされていれば、今度はこの記述を取り出し、同じ様に処理ループを続ける。

【００７７】３．３　処理負荷の見積り動作（図７）

負荷の検出方法については、既に「２．１　本実施例の負荷計算手段」で述べた。ここでは、処理時間算定手段１を呼び出す機械語命令EVAL[2f]の動作を図７によって説明する。なお本実施例では仮想機械語命令としてEVAL[2f]を定義し使用したが、この命令自体は複数の機械語の組み合わせでマクロ定義できる。以下は、負荷見積りの命令語を実装する上での例を説明するものである。

【００７８】EVAL処理（Ｓ６０３）は、分散されたＰＥ２００上では、内部フラグ＝１であるため実行されない。これ以外のとき、仮想機械プロセス４０３は、負荷カウンタ２を初期化し（Ｓ７０１）、続く機械語命令をフェッチする（Ｓ７０２）。この機械語命令が、TRAP[2a]であるとき、前述の処理時間算定手段１を実行する（Ｓ７０３）。この処理結果は、負荷カウンタ２に加算される（Ｓ７０４）。その他の命令は、通常処理される。しかし仮想機械プロセス４０３は、EXIT[2d]命令を

検出した場合、処理分散手段４を呼び出し、既にFORK[2c]命令によって複製されていたユーザ記述を、他のＰＥ２００に転送する（Ｓ７０５）。この時、処理負荷によって適切なプロセッサが選択される必要がある。この処理は処理分散手段４の特徴であり、既に図５を用いて述べた通りである。

【００７９】４．他の実施例（図１５）

再び図４をに戻り説明する。エディタ４０１とコンパイラ４０２は、必ずしも本実施例の様にマルチプロセッサ画像処理装置の上で実行される必要は無い。４０１、４０２を他のコンピュータで実行し、その処理結果として得られるユーザプログラム４０４を、マルチプロセッサ画像処理装置への入力として与えても良い。この時、マルチプロセッサ画像処理装置側には、少なくとも仮想機械プロセス４０３を実装する必要がある。また、処理時間算定手段１を含んだ画像処理ライブラリ７２が必要である。

【００８０】図１５（ａ）は、本発明の別の実施例として挙げる印刷装置の構成図である。パーソナルコンピュータ１５１と通信経路１５４で接続した電子写真式のプリンタにおいて、マルチプロセッサ画像処理装置１５３が画素発生を行ない、処理結果をプリントエンジン１５２に出力する。この構成を更に図１５（ｂ）を用いて説明する。

【００８１】コンピュータ１５１の使用者は、アプリケーション１５５を使用し、ドキュメントの印刷操作を行なう。この時、アプリケーション１５５はＯＳ１５９の用意するプリンタドライバ１５７を起動し、印刷処理を行なう。プリンタドライバ１５７はアプリケーションの指示に従い、画像記述のソースコードを生成し、続いてコンパイルまでを行なう。この結果、プリンタドライバ１５７は前の実施例に述べたような仮想機械の命令語からなるデータを作成する。次にプリンタドライバ１５７が、このデータをデバイスドライバ１５８を用いて通信経路１５４に送る。このデータはプリンタに組み込まれたマルチプロセッサ画像処理装置１５３によって受信される。

【００８２】マルチプロセッサ画像処理装置１５３内の仮想機械プロセス４０３は、画像処理ライブラリ７２を利用して前記実施例と同様に処理を行ない、処理結果をプリントエンジン１５２に出力する。

【００８３】以上２つの実施例に示した様に、本発明では負荷を見積もってからプロセッサに分散する手段が、最適負荷分散を実現している。そこで、本発明の方法を、各プロセッサがより独立して構成された処理装置に用いる事が考えられる。前記実施例では、プロセッサ間の通信をシステムバス３５によって行なった。これに替えて、プロセッサ間の通信をローカルエリアネットワーク（以下ＬＡＮ）によって行なう事が考えられる。この方式を用いた場合、ＬＡＮによって接続された複数のワークステーション（以下ＷＳ）を用意し、この上で前記実施例のと等価な仮想機械プロセスを実行する。ＷＳはマルチタスク処理を行なうＯＳを実装しており、ここに仮想機械プロセスを実装した場合、システムバス３５に替えてＬＡＮを用いる以外は、ほぼ前記実施例と同様の処理が可能である。この構成では、複雑なドキュメントの画像処理／印刷処理を、適当な負荷配分の元に複数台のＷＳを用いて処理できるため、処理速度面で極めて優秀な処理装置を実現できる。

【００８４】

【発明の効果】以上の説明から明らかな様に、本発明のマルチプロセッサ画像処理装置では、実行時ライブラリ中の操作の一つ一つに対応した処理負荷の見積り手段が備わっている事により、画像処理の実際の操作開始に先立って処理負荷を見積もる事ができるため、現在待ち行列にある処理要求も含めて個々のプロセッサの負荷を正しく評価する事が可能となった。これにより、本発明ではプロセッサに多大な負荷を与えるため分割して処理を行なう事が不適当な画像の拡大・縮小、回転等の画像処理分野において、各プロセッサへの処理の分散を適切な負荷配分の下に行なうことに成功した。

【００８５】言い替えれば、本発明は個々のプロセッサへの負荷配分を均等化する有効手段を提供するものであり、いわゆる「粒度の大きな」並列処理において、マルチプロセッサ装置の性能を最大限に引き出すことを可能にする効果が有る。

【図面の簡単な説明】

【図１】　第１の実施例の特徴となる構成を示す図。

【図２】　プログラムの実行時の状態の説明図。

【図３】　マルチプロセッサ画像処理装置の全体の構成を示す図。

【図４】　マルチプロセッサ画像処理装置の処理手順の概要を説明した図。

【図５】　処理分散手段４の動作を説明した流れ図。

【図６】　仮想機械プロセスの動作を説明した流れ図。

【図７】　機械語EVALの動作を説明した流れ図。

【図８】　機械語EXIT及びWAITの動作を説明した流れ図。

【図９】　直線発生時の負荷見積り方法を示した流れ図。

【図１０】　画像回転の例を説明する図。

【図１１】　実施例で用いた並列記述言語の構文規則の説明図。

【図１２】　並列記述言語による記述例の説明図。

【図１３】　図１２の記述例をコンパイルした時の機械語の説明図。

【図１４】　粒度のばらつきの大きい場合の負荷分散の問題点を説明した図。

【図１５】　第２の実施例の構成図。

【符号の説明】

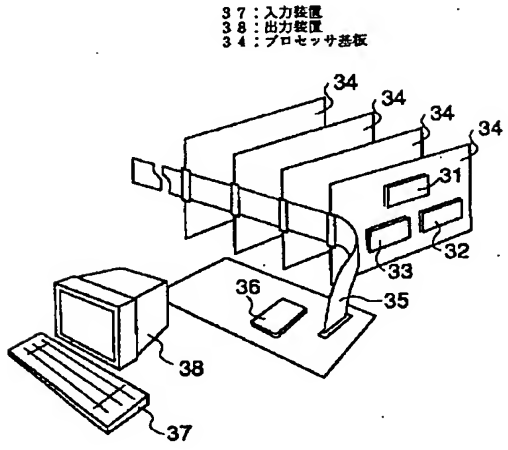
17

１…処理時間算定手段
２…負荷カウンタ
３…プロセッサ管理テーブル
４…処理分散手段
５…ユーザの記述によるオブジェクトコード
６…仮想機械プロセス
７…仮想機械インタープリタ
８…システムの資源
３１…ＭＰＵ
３２…ＲＯＭ
３３…ＲＡＭ
３４…プロセッサ基板
３５…システムバス
３６…Ｉ／Ｏ装置
４１…待ち行列
４２…処理要求
７１…プログラムインターフェース
７２…画像処理実行時ライブラリ
８１…通信手段
８２…待ち行列
１００…図形
１０１…座標
１０２…回転した座標
１０３…図形を囲む領域
１２１…cobegin文
１２２…coend文

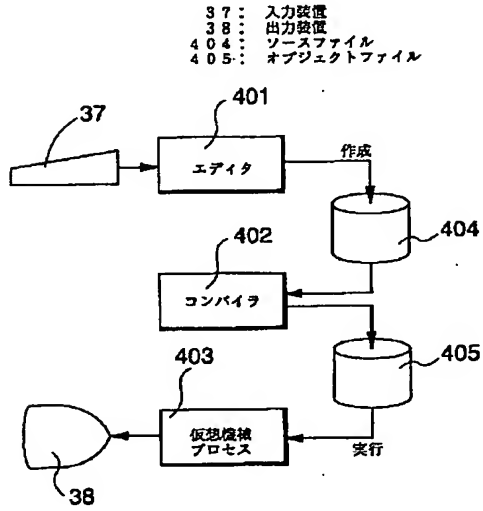
18

＊１２３…並列記述文
１２４…otherwise文
１３５…翻訳された機械語の例
１５１…パーソナルコンピュータ
１５２…プリントエンジン
１５３…マルチプロセッサ画像処理装置
１５４…通信経路
１５５…アプリケーション
１５７…プリンタドライバ
１５８…デバイスドライバ
１５９…オペレーティングシステム
２００…一つのプロセッサエレメント
２０１…プロセス
２０２…オブジェクトコード
２０３…バス調停装置
２０４…オペレーティングシステム
２０５…スケジューラ
２０６…スケジューラ待ち行列
２０７…サービスプログラム
４０１…エディタ
４０２…コンパイラ
４０３…仮想機械プロセス
４０４…ソースファイル
４０５…オブジェクトファイル
７１１…待ち行列
＊　７１２…分岐テーブル

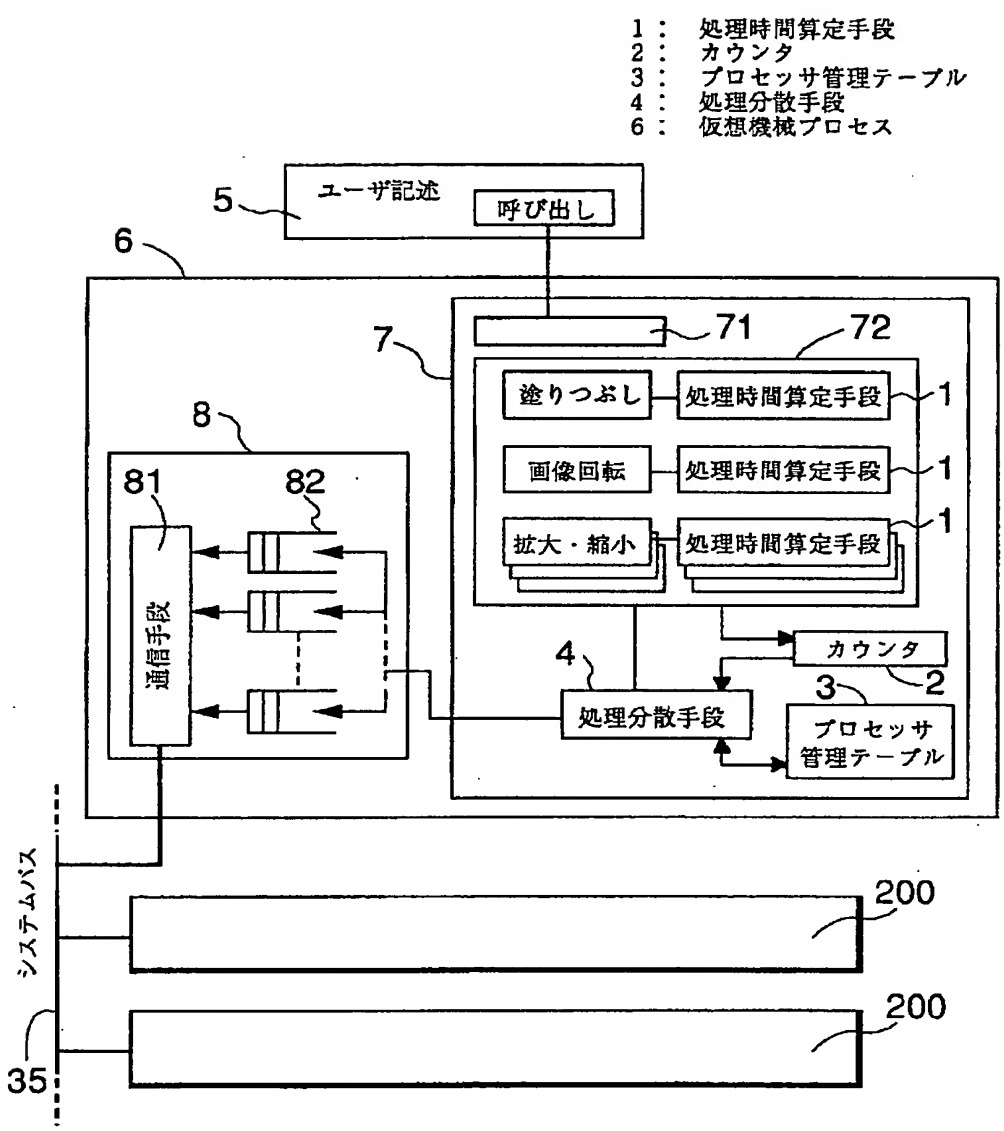
【図３】



【図４】

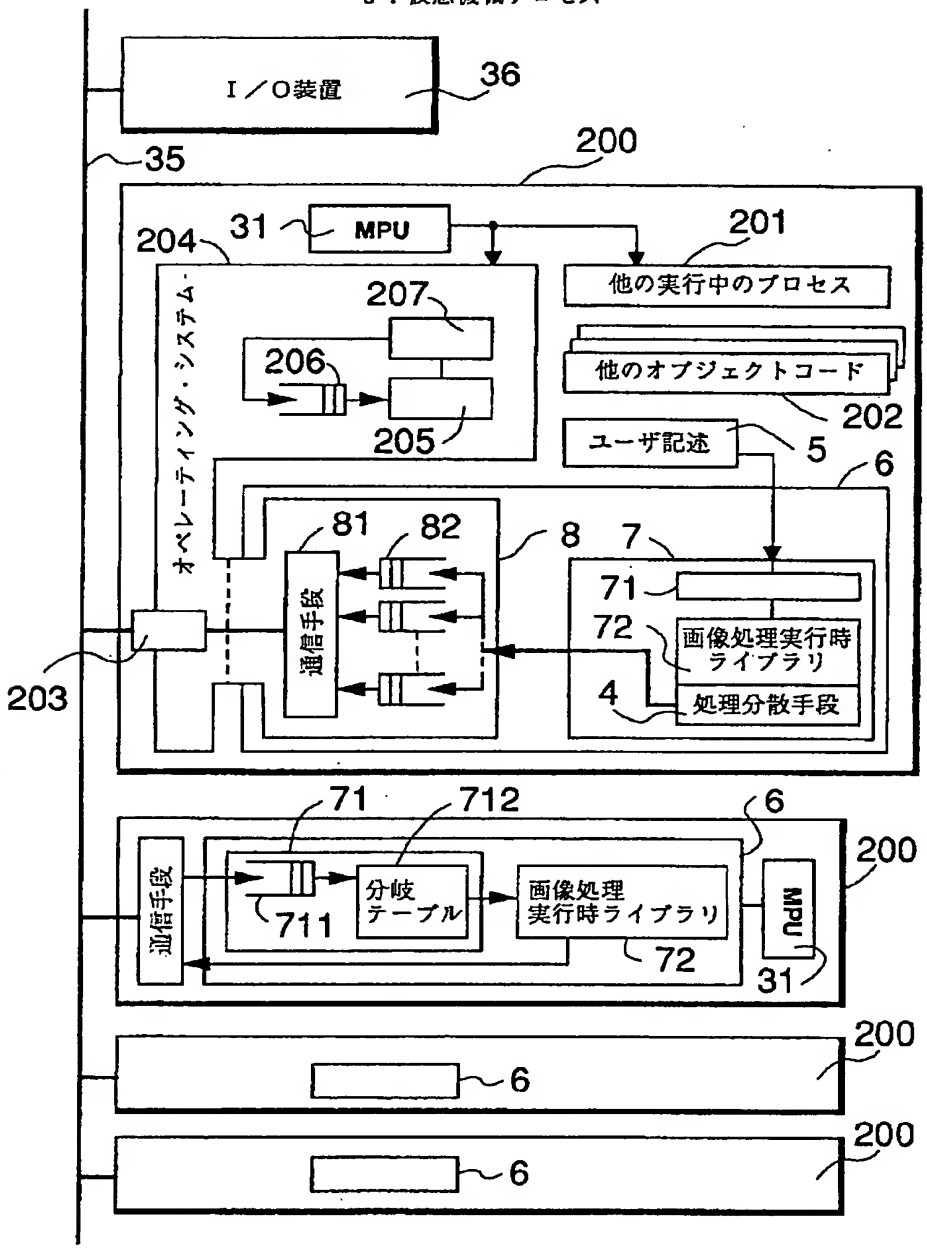


【図１】

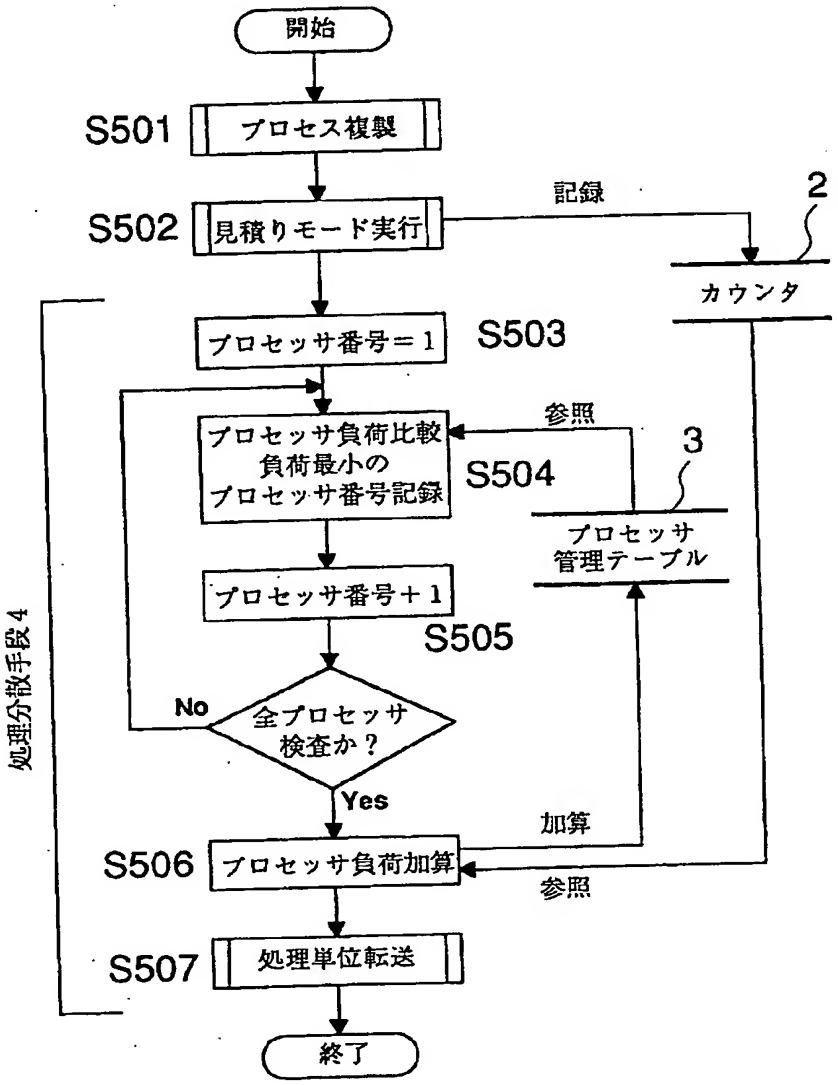

1：　処理時間算定手段
2：　カウンタ
3：　プロセッサ管理テーブル
4：　処理分散手段
6：　仮想機械プロセス
5
ユーザ記述
呼び出し
6
7
71
72
途りつぶし
処理時間算定手段
1
画像回転
処理時間算定手段
1
拡大・縮小
処理時間算定手段
1
8
81
82
通信手段
カウンタ
2
4
3
処理分散手段
プロセッサ
管理テーブル
システムバス
200
200
35

【図２】

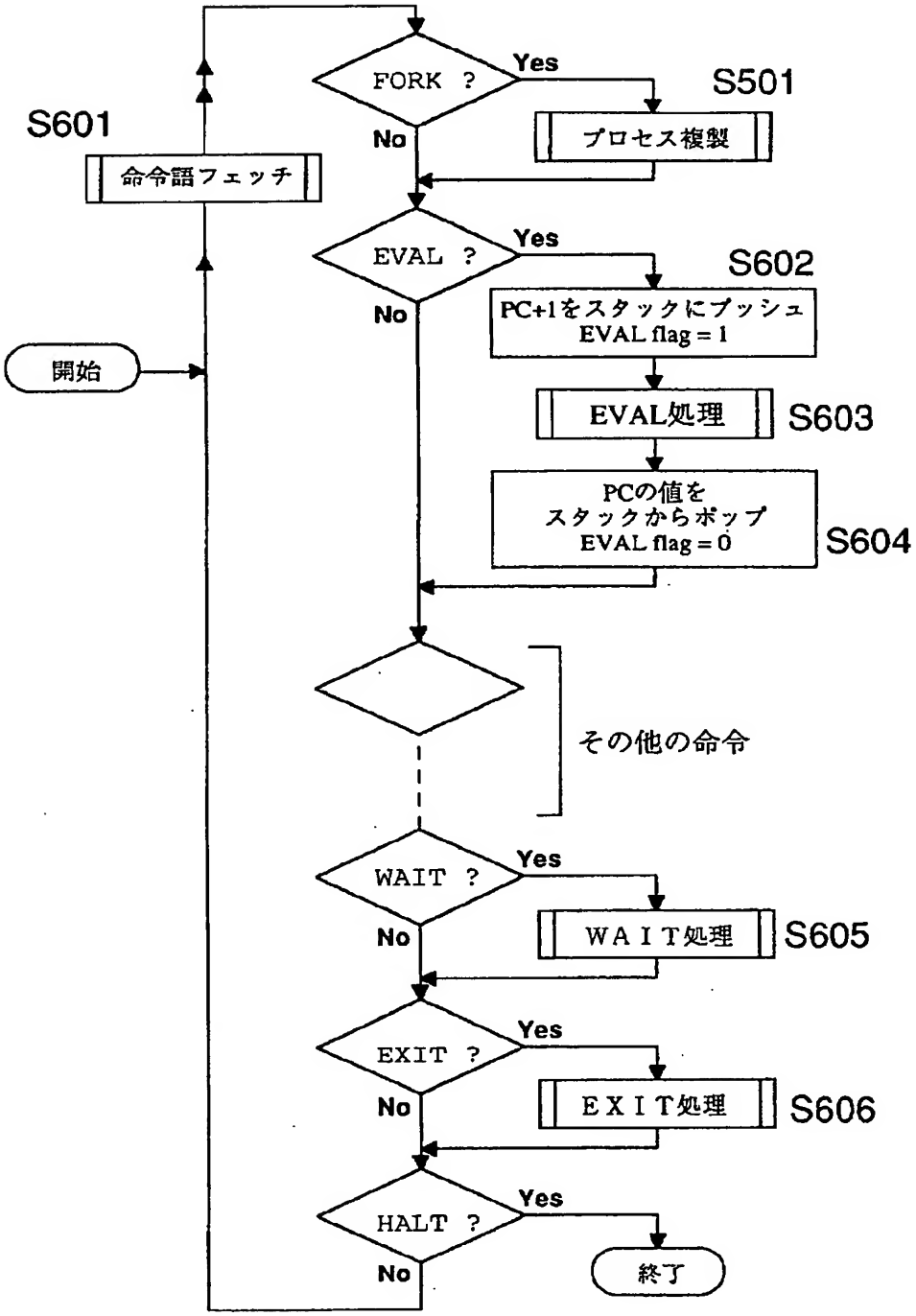
２００：　１つのプロセッサ基板上のシステム
（プロセッサ・エレメントと呼ぶ）
５：並列記述から生成したオブジェクトコード
６：仮想機械プロセス



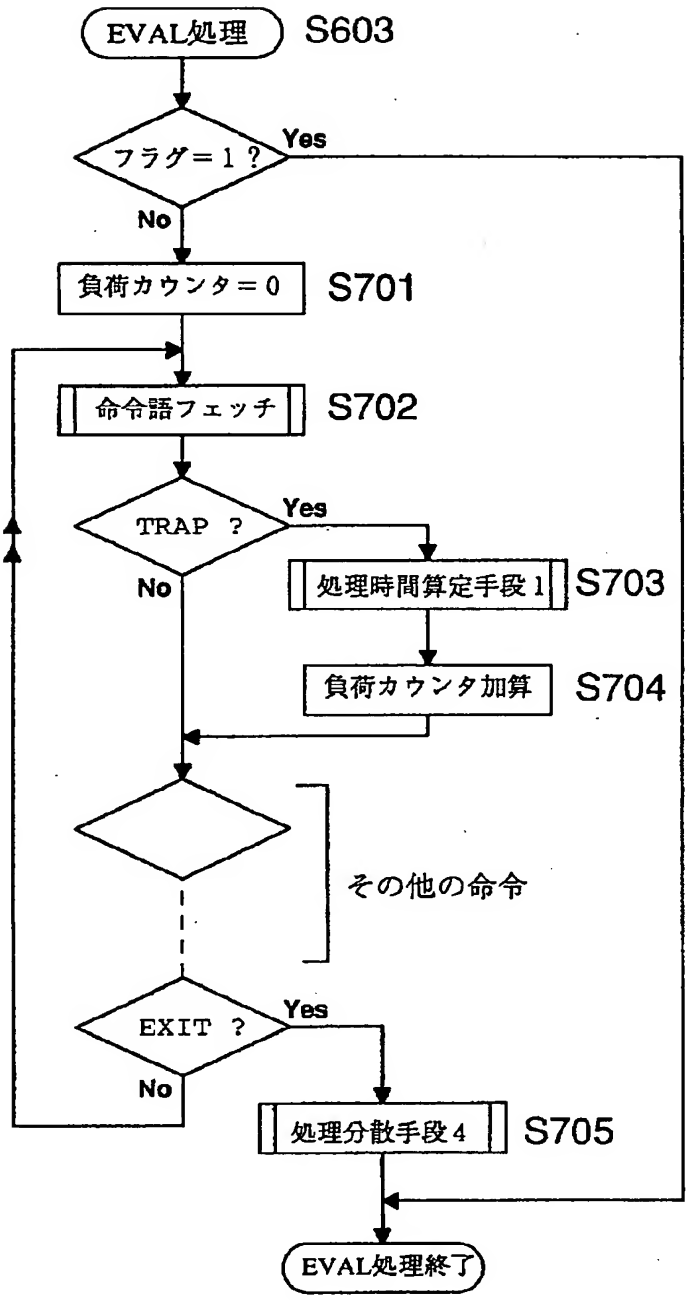
【図５】


開始
S501
プロセス複製
S502
見積りモード実行
記録
2
カウンタ
プロセッサ番号＝1
S503
参照
プロセッサ負荷比較
負荷最小の
プロセッサ番号記録
S504
3
プロセッサ
管理テーブル
プロセッサ番号＋1
S505
No
全プロセッサ
検査か？
Yes
加算
S506
プロセッサ負荷加算
参照
S507
処理単位転送
終了
処理分散手段４

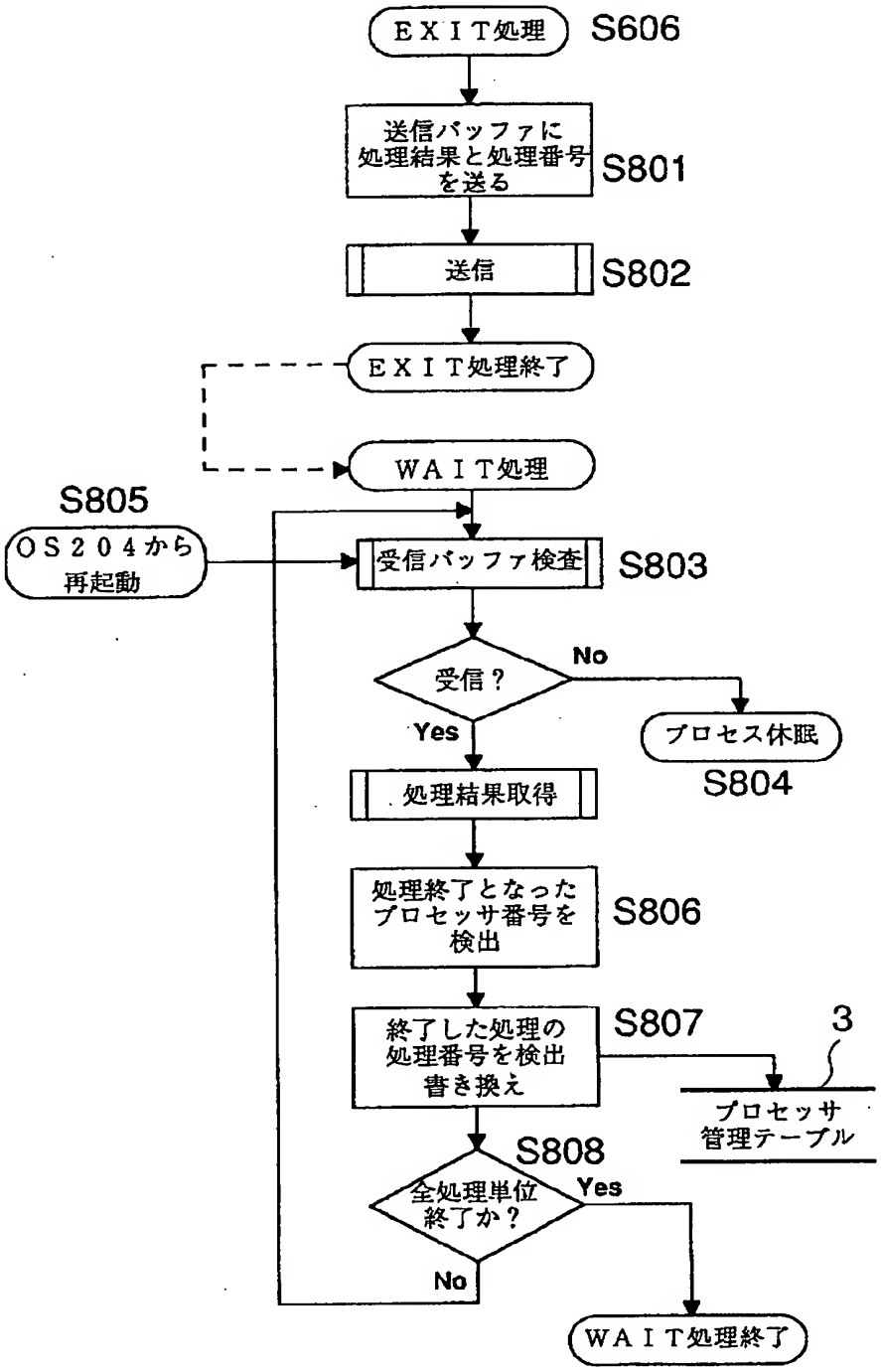
【図６】


開始
FORK ?
Yes
S501
プロセス複製
No
S601
命令語フェッチ
EVAL ?
Yes
S602
No
PC+1をスタックにプッシュ
EVAL flag = 1
EVAL処理
S603
PCの値を
スタックからポップ
EVAL flag = 0
S604
その他の命令
WAIT ?
Yes
No
WAIT処理
S605
EXIT ?
Yes
No
EXIT処理
S606
HALT ?
Yes
No
終了

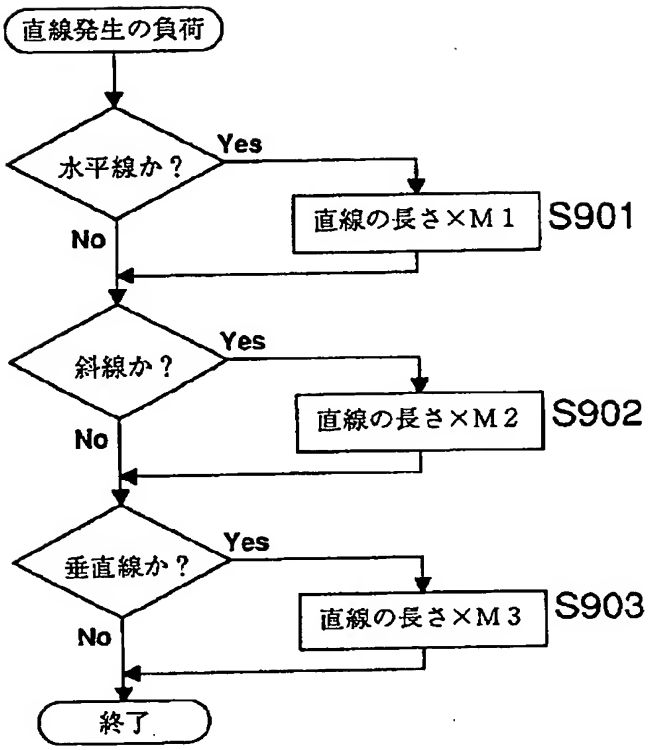
【図７】


EVAL処理
S603
フラグ＝１？
Yes
No
負荷カウンタ＝０
S701
命令語フェッチ
S702
TRAP ？
Yes
No
処理時間算定手段１
S703
負荷カウンタ加算
S704
その他の命令
EXIT ？
Yes
No
処理分散手段４
S705
EVAL処理終了

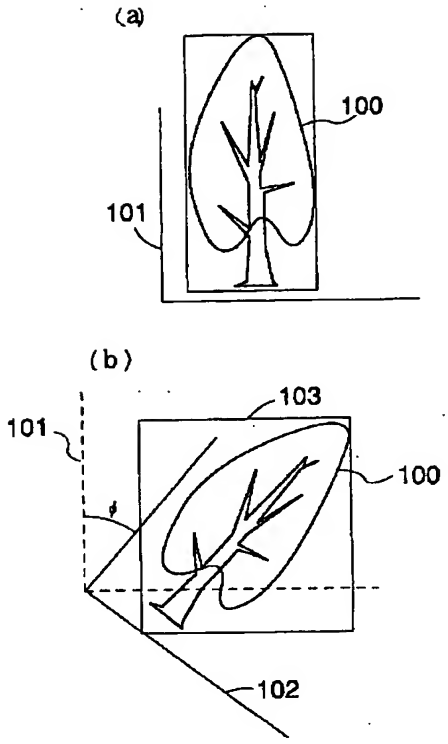
【図８】


ＥＸＩＴ処理
S606
送信バッファに処理結果と処理番号を送る
S801
送信
S802
ＥＸＩＴ処理終了
ＷＡＩＴ処理
S805
ＯＳ２０４から再起動
受信バッファ検査
S803
受信？
No
Yes
プロセス休眠
S804
処理結果取得
処理終了となったプロセッサ番号を検出
S806
終了した処理の処理番号を検出書き換え
S807
3
プロセッサ管理テーブル
S808
全処理単位終了か？
Yes
No
ＷＡＩＴ処理終了

【図９】



【図１０】



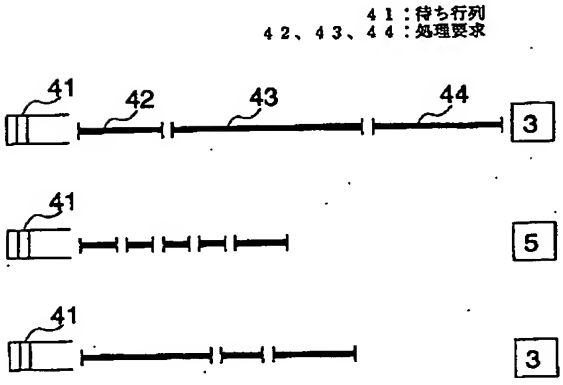
【図１２】

```
program sample;
  var
      x1,x2,y1,y2 : integer;

  procedure user1(a,b : integer);
    begin
        ......
    end;

  begin
    cobegin  ----------------------------121
      // line(20,20,120,220);
123 ~~  // begin
123 ~~      stroke(3,10,10,80,60,40,90); fill;
          end;
123 ~~  // user1(x1,100);
124 ~~  otherwise
          x1:=10; y1:=20;
    coend;  ----------------------------122
  end.
```

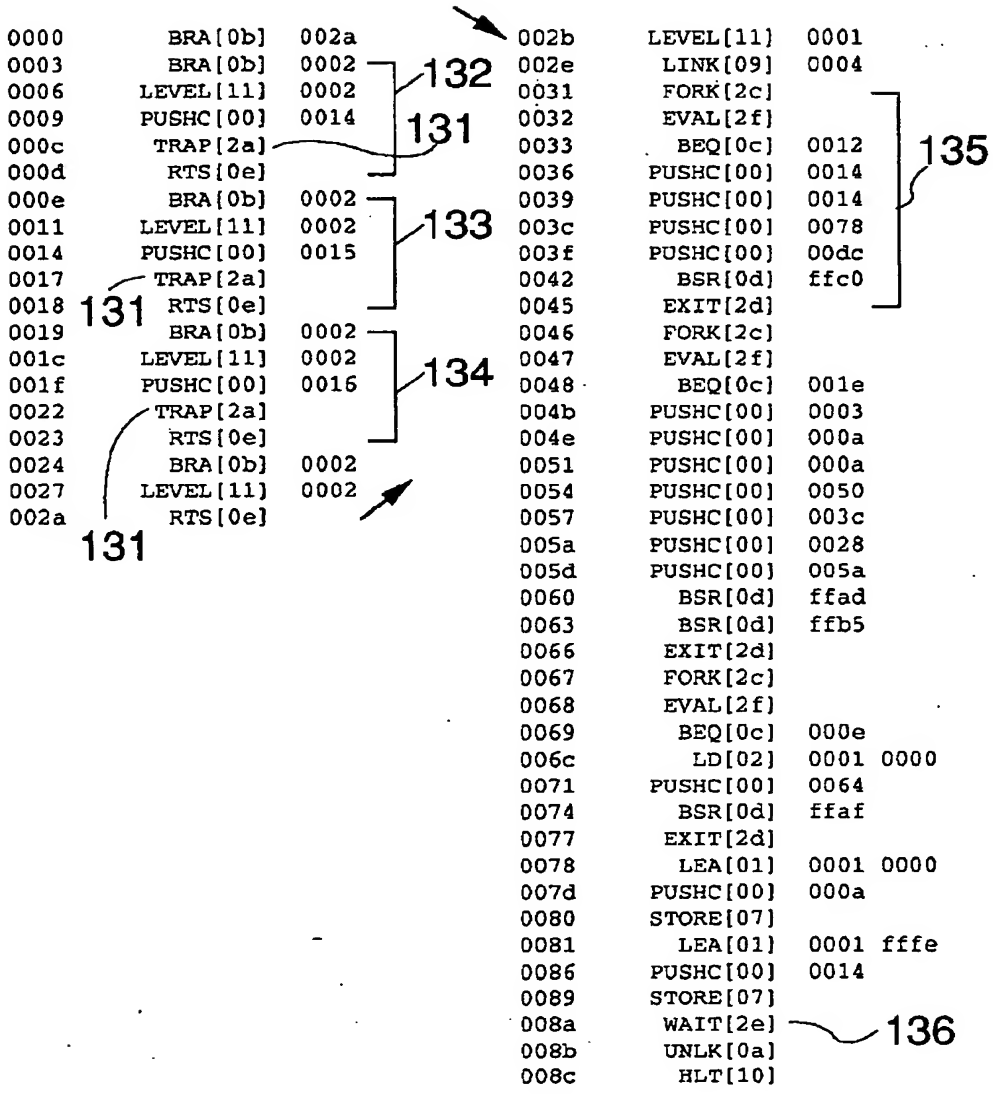
【図11】

```
compound_stm
 : BEGIN statements END
 ;
statements
 : statement
 | statements ';'
 | statements ';' statement
 |
 ;
statement
 : p_f_call
 | variable ASSIGN expression
 | compound_stm
 | IF expression THEN statement opt_else
 | WHILE expression DO statement
 | REPEAT statements UNTIL expression
 | FOR IDENT ASSIGN expression direction expression DO statement
 | COBEGIN para_body opt_other COEND
 ;
opt_else
 : ELSE statement
 |
 ;
para_body
 : para_stm
 | para_stm ';'
 | para_stm ';' para_body
 ;
para_stm
 : "//" statement
 ;
opt_other
 : OTHERWISE statements
 |
 ;
```

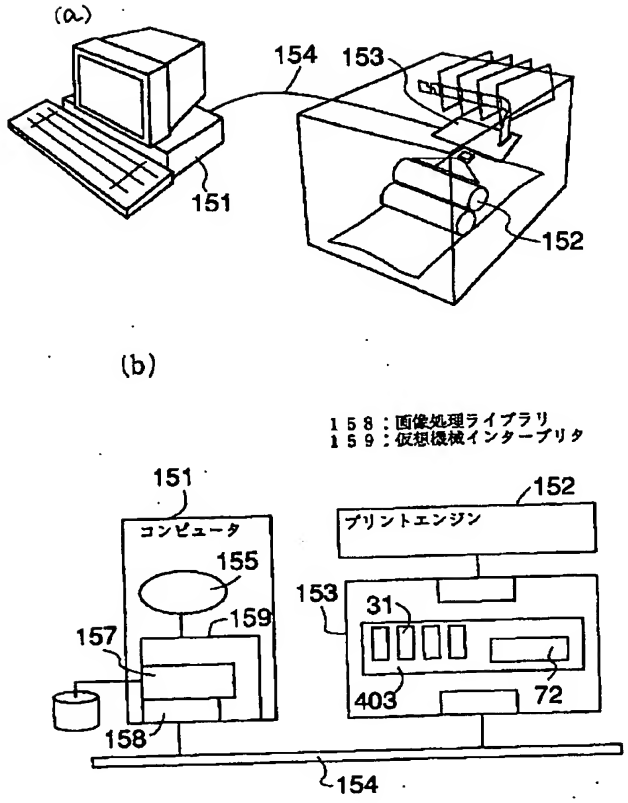
【図14】



【図13】


0000 BRA[0b] 002a
0003 BRA[0b] 0002
0006 LEVEL[11] 0002
0009 PUSHC[00] 0014
000c TRAP[2a]
000d RTS[0e]
000e BRA[0b] 0002
0011 LEVEL[11] 0002
0014 PUSHC[00] 0015
0017 TRAP[2a]
0018 RTS[0e]
0019 BRA[0b] 0002
001c LEVEL[11] 0002
001f PUSHC[00] 0016
0022 TRAP[2a]
0023 RTS[0e]
0024 BRA[0b] 0002
0027 LEVEL[11] 0002
002a RTS[0e]
002b LEVEL[11] 0001
002e LINK[09] 0004
0031 FORK[2c]
0032 EVAL[2f]
0033 BEQ[0c] 0012
0036 PUSHC[00] 0014
0039 PUSHC[00] 0014
003c PUSHC[00] 0078
003f PUSHC[00] 00dc
0042 BSR[0d] ffc0
0045 EXIT[2d]
0046 FORK[2c]
0047 EVAL[2f]
0048 BEQ[0c] 001e
004b PUSHC[00] 0003
004e PUSHC[00] 000a
0051 PUSHC[00] 000a
0054 PUSHC[00] 0050
0057 PUSHC[00] 003c
005a PUSHC[00] 0028
005d PUSHC[00] 005a
0060 BSR[0d] ffad
0063 BSR[0d] ffb5
0066 EXIT[2d]
0067 FORK[2c]
0068 EVAL[2f]
0069 BEQ[0c] 000e
006c LD[02] 0001 0000
0071 PUSHC[00] 0064
0074 BSR[0d] ffaf
0077 EXIT[2d]
0078 LEA[01] 0001 0000
007d PUSHC[00] 000a
0080 STORE[07]
0081 LEA[01] 0001 fffe
0086 PUSHC[00] 0014
0089 STORE[07]
008a WAIT[2e]
008b UNLK[0a]
008c HLT[10]
131
132
131
133
134
131
135
136

【図１５】


(a)
154
153
151
152
(b)
１５８：画像処理ライブラリ
１５９：仮想機械インタープリタ
151
コンピュータ
155
159
157
158
152
プリントエンジン
153
31
403
72
154